8) Polymorphism: Run time

↓
Take decision at run time.

Vehicle *v2 = new car;
v2 → print(); → vehicle()

v2 → print() ~~acc~~ → To get car class print() fun this we need

* To achieve Run time polymorphism we can use Virtual functions.

```
Class Vehicle {
  public:
    string color;
  virtual void print() {
        cout << "Vehicle" << endl;
  }
};

class car : public Vehicle {
   public:
      int num Gears;
   void print() {
      cout << 'Car' << endl;
   }
};
```

Now when we print

~~v_2 print~~ Vehicle* v2 = new car;

v2 -> print() -> now its ~~[illegible]~~ call car class print

* If print() function is not defined in car class then automatically the Base ~~[illegible]~~ class ~~[illegible]~~ print function is called.

* If print function is not defined in vehicle, then it raises an error.

---

* Run time polymorphism is achieved using virtual functions

<u>Use case of Runtime polymorphism</u>

i) Organization:

Calculating salaries:

$$\left.\begin{array}{l} \to \text{HR} \\ - \text{Manager} \\ - \text{Engineers} \\ - \text{Others} \end{array}\right\} \text{these have different salaries}$$

$$\begin{tikzcd}
 & & \text{Employee} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{HR} & \text{Manager} & \text{Engineering} & \text{Others} \\
(9) & (3) & (20) & (4)
\end{tikzcd}$$

→ Every child class has a function called calculateSalary().

Employee **c ~~= new Employee*[28]~~

new Employee * [28]